Geol 335.3

Lab #5: Reflection Seismic Synthetics

In this lab, you will write a Matlab program to study reflections from a relatively complexly shaped reflecting interface. The program will also be used to produce a zero-offset section that will be inverted in the next lab.

Theory

To model reflection responses from an arbitrary boundary, we can represent the boundary by a sequence of closely spaced reflection (or "scattering") points. In this way, there is really no difference between linear segments and points at which the reflecting boundary is interrupted or changes direction. Reflections from continuous linear segments will add up coherently and form the usual reflection hyperbolas. Reflections from isolated scattering points, kinks, and edges of reflectors with provide *diffractions*.

In the program, we will represent the elements of the model as follows:

To represent:	*Use:*
Coordinates and vectors: (x,z)	2-element arrays `[x,y]`
Reflectors	2-dimensional arrays (e.g., matrices; arrays of coordinates)
Time series	1-dimensional arrays
Output data sections	2-dimensional array (arrays of time series, each associated with a surface location)

Accordingly, the problem can be treated and encoded in Matlab with the use of vectors and array operations.

Code

Try subdividing the code into a set of functions with well-defined functionalities. Common parameters, such as the model and receiver geometry, can be provided in common data space using the `global` declaration in Matlab:

```
global      receivers   % receiver locations
global      sources     % source locations
global      velocity    % velocity
global      layer       % array of reflecting interface points
global      time        % time sampling
global      xref wref   % source waveform
```

Here, array `receivers` should contain a list of receiver positions, and array `time` – list of times at which the resulting records are sampled. `Velocity` is the overburden velocity. These variables should of course be set before calling any function using them.

The following functions from the web site are required:

1. Model setup, principally containing the global statements above; this script is provided in file `model.m`;
2. Interactive picking of a reflective interface (this function `picklayer` is provided);
3. Function `reflection` returning the reflection section resulting from a single boundary.
4. Function `plot_section` performing plotting of the resulting seismic sections.
5. From Lab #4, function `wavelet(…)` and file `wavelet.dat` required by for it.

To begin, look into the header of each of these functions and understand the meanings of their parameters, return values, and algorithms. In your own code, make sure to document processing by similarly providing commentaries describing the parameters, assumptions, and return values.

Assignments

1. Copy model.m into a new file model1.m and modify it by making calls to calculate a seismic reflection sections resulting from each of the three sources. These sections are obtained like this (for source #1):

```
sec = reflection(sources(1),receivers,layer,2.0);
plot_section(receivers,sec,'Distance (m)','b-')
```

   Plot each of the three shorts in different colors in the same (or different, as you find better) plot.

2. Execute your script `model1.m` and pick a reflecting boundary (2-3 points) with a reflector dipping to the right. Make (save) a plot and save the model and the resulting record section in Matlab namespace `model1.dat`.
   a. Describe the differences in the sections from different sources (located up- and down dip);
   b. Save the plots and include them in the report.
3. Create two additional copies of your model `model2.m` and `model3.m`, in which you pick more complex reflecting boundaries (5-8 points) containing an approximately horizontal segment with an uplift (model2) and a depression (model3). Repeat the above steps and make observations of the reflections.
4. Make another copy `model4.m` and modify it to create a synthetic "zero-offset section" (ZOS). The zero-offset section is produced by collocated sources and

receivers. (one receiver for each shot). This can be achieved by simply using receiver coordinates as sources:

```
sec = reflection(receivers,receivers,layer,2.0);
```

However, note that this computation takes about 14 times longer (there are many more sources this time). If the wait time is impractical, increase the spacing between receivers in this model.

a. Compare the results of the ZOS model to the preceding ones.

b. Compare the shapes of reflectors in the ZOS to the actual reflectors you created in `picklayer()`

c. Save the plots and include them in the report.

Hand in:

Zipped directories containing:

1. All Matlab codes (“m-files”);
2. Screen captures or Postscript/PDF figures;
3. Summary and discussion in a Word file.